Convolutional Bilateral Multi-Perspective Matching for Natural Language Inference

Natural Language Processing Final Project Report

Jinning Li, 515030910592
Shanghai Jiao Tong University
lijinning@sjtu.edu.cn

1 Introduction

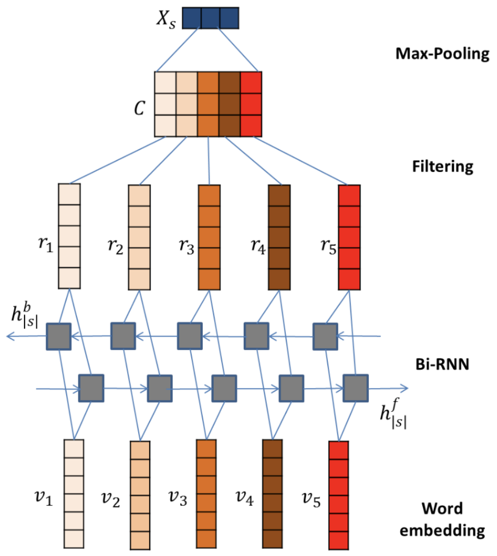
Natural Language Inference (NLI) task aims to determine the logical relationship between two sentences (a premise and a hypothesis). **entailment**: if the premise is true, then the hypothesis must be true. **contradiction**: if the premise is true, then the hypothesis must be false. **neutral**: neither entailment nor contradiction

The Stanford Natural Language Inference (SNLI) corpus is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels. In this project, all my experiments will be based on the SNLI corpus.

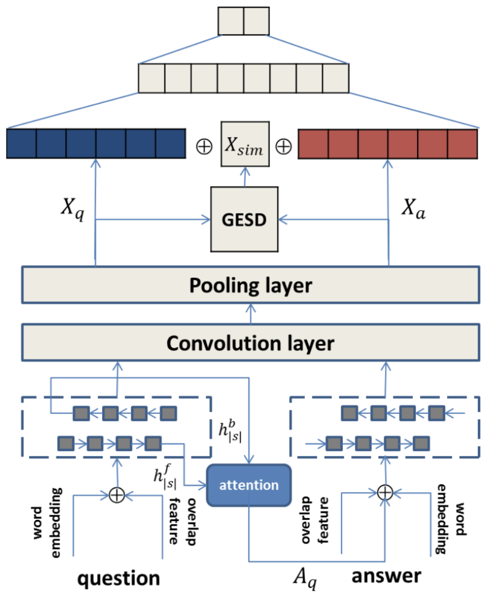
The hybrid framework [5] of *Convolutional Recurrent Neural Network* (convRNN) is a new consideration to encode the sentences and capture the interaction between them. In my project, I want to try some sentences encoding based and Convolutional Neural Network based models. So the hybrid of CNN and RNN becomes my basic framework. Based on this ideas, I first build a hybrid architecture, namely, *ConvRNN-Naive*. In the beginning, the sentences of premises and hypothesis are encoded by an word embedding layer. After that, a Bi-directional Long-Short Term Memory (Bi-LSTM) layer is applied to encode the sentences both forwards and backwards. This is because the Bi-LSTM is capable of capturing the overall information of a sentence. Then, a convolutional layer and a max-pooling layer is applied to find the local n-gram features in a sentence. In the end, I use a fully connected layer to make the classification. The ConvRNN-Naive only achieves an accuracy of 79.0% in my experiment, which is not so satisfying.

However, the ConvRNN-Naive model performs not so well. So, inspired by the ideas of skip-connection in U-Net [3] and highway networks in [4], I use the hidden states of P and H in Bi-LSTM layer to build a interaction between them. I then feed the interaction vector and hidden state vectors to the fully connected layer. This is just like a short-cut which avoids the convolutional encodings. Experiments prove that this model achieves an accuracy of 83.3% on test set. This architecture is named as *ConvRNN with Skip-connection* (ConvRNN-Skip).

The *bilateral multi-perspective matching* (BiMPM) model [6] achieves a high performance on the SNLI dataset. Given two sentences P and Q, BiMPM first encodes them with a Bi-LSTM encoder. Next, it match the two encoded sentences in two directions P against H and H against P. In each matching direction, each time step of one sentence is matched against all timesteps of the other sentence from multiple perspectives. Then, another Bi-LSTM layer is utilized to aggregate the matching results into a fixed-length matching vector. Finally,



(a) The architecture of ConvRNN for single sentence classification



(b) The Architecture of ConvRNN-Naive for natural language inference

Figure 1: Architecture of ConvRNN and ConvRNN-Naive

based on the matching vector, a decision is made through a fully connected layer. This work inspires me to use the matching between P and H to better capture the relationshp between them. An accuracy of 86.4% is achieved by BiMPM.

Based on the idea of BiMPM, I propose a new architecture named *Convolutional BiMPM* (CBiMPM). The sequences of P and Q produced by Bi-LSTM layer are designed to go through a convolutional layer and a max-pooling layer. Then, a convolutional interaction between them is applied. The output of max-pooling layer and the interaction vecter are combined with the result of BiMPM before the fully connected layer. Experiments prove CBiMPM achieves a best accuracy of 86.7% under the same hyper-parameters and experiment environment.

2 Implements

2.1 ConvRNN-Naive

The architecture of my first architecture convRNN-Naive is shown in Fig.1b. The basic convRNN includes word embedding layer, Bi-RNN layer, convolutional filtering, max-pooing, and classification layers.

Word embedding layer Thee original input is a sentence S consisting of a sequence of words: $[w_1, ..., w_{|s|}]$, where each word is drawn from a finite-sized vocabulary V with size $|V|$. Before fitting into the next layer, each word is transformed into a low-dimensional dense vector via a lookup table operation: $i = LT_W(w_i)$, where $W \in R^{dw \times |V|}$ is the word embedding matrix and d_w is each word embedding dimension. As a result, the input sentence S is represented as a matrix where each column corresponds to a word embedding.

BI-RNN layer The sentence matrix is then fitted into the BI-RNN layer with dimension d_r. Single directional RNN is insufficient and suffers from not utilizing the contextual information from the future words. The BI-directional RNN utilizes both previous and future contexts by processing the sequence on both forward and backward directions. At each time step t, the output r_t is the concatenation of the two output vectors r_t^f and r_t^b from both directions. Besides, the final hidden states $h_{|s|}^f$, $h_{|s|}^b$ from both directions are usually used for representing the whole sentences.

Convolution layer Given the output of BI-RNN layer, $R \in R^{|s| \times 2d_r}$, the convolution layer uses a set of n filter vectors $f_i \in R^{2md_r}$ with sliding window size m to process it by a linear convolution operation. Formally, let $R_{i:i+j}$ refer to the concatenation of $r_i, r_{i+1}, ..., r_{i+j}$. The linear convolution operation takes the dot product of f_i with each m-gram in the sentence S shown as follows:

$$C_{it} = f_i^T \cdot R_{t-m+1:t} \tag{1}$$

Poolinglayer Pooling, including max-pooling, min-pooling and average-pooling, is usually used to extract robust features from the results of convolution operation. In this paper, we propose to apply max-pooling for each filter to capture the most significant signal. Formally, we extract the max value from each row of C, which will generate the final representation vector $X_s \in R^n$ for the input sentence S

Then in the Natural Language Inference, the only difference of ConvRNN-Naive is the convRNN process is applied to both the premises P and hypothesis H. And an interaction between them should be build to help the neural network better capture the interaction between P and H. In my project, the the Geometric mean of Euclidean and Sigmoid Dot (GESD) is used to measure the relatedness between the two representations:

$$X_{sim} = \frac{1}{1 + \|x - y\|} \times \frac{1}{1 + \exp(-\lambda(xy^T + c))} \tag{2}$$

Then, the vector X_P, X_H, and X_{sim} is fitted into the fully conncected layer and make a final classification.

2.2 ConvRNN-Skip

The idea of ConvRNN-Skip is just to add skip hidden-states and their interaction into the classification layer. The hidden states of P, Q are denoted as h_P and h_Q. Then, a pair-wise product is used to build the similarity vector h_{sim}, $h_{sim} = h_P \times h_Q$. The architecture of ConvRNN-Skip is shown in Fig.2.

2.3 BiMPM

The kernel idea of *bilateral multi-perspective matching* (BiMPM) is the matching process, which is shown as interactive curve between the left side and the right side in Fig.3a. The goal of the Matching layer is to compare each contextual embedding (time-step) of one sentence against all contextual embeddings (time-steps) of the other sentence. As shown in

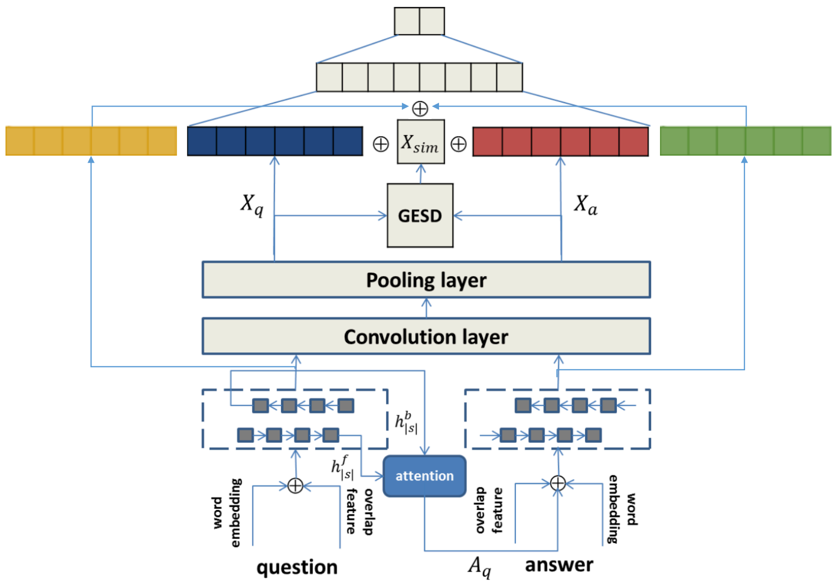


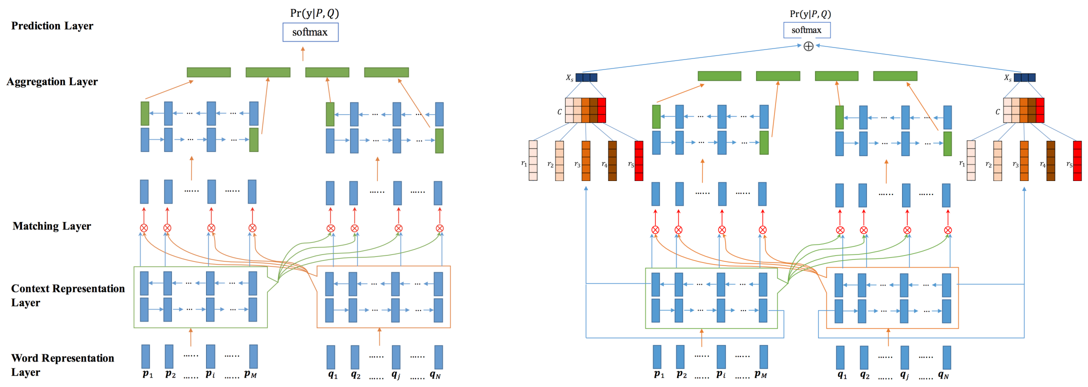
Figure 2: Architecture of ConvRNN-Skip

Fig.3a, we will match the two sentences P and H in two directions: match each time-step of P against all time-steps of H, and match each time-step of H against all time-steps of P. To match one time-step of a sentence against all time-steps of the other sentence, we design a multi-perspective matching operation $\otimes$. The output of this layer are two sequences of matching vectors (right above the operation $\otimes$ in Fig.3a), where each matching vector corresponds to the matching result of one time-step against all time-steps of the other sentence.

2.4 Convolutional BiMPM

The main difference of Convolutional BiMPM is that the sequences of P and Q produced by Bi-LSTM layer are designed to go through a convolutional layer and a max-pooling layer. Then, a convolutional interaction between them is applied by a pair-wise product.

In the convolutional layer, the kernel size is set as $n \times 2|h|$, where n is the length of disired local n-gram. $2|h|$ here is the hidden size of both directions. After the convolutional layer, the data becomes $batch \times channels \times seq\_len$, where $channels$ is the number of convolutional feature map. seq_len is the length of a input sentence. In the max-pooling layer, the dimension of seq_len is squeezed by maximum operation. The output become $batch \times channels$, denoed as X_P and X_H. Then, a pairwise product is applied to get the interaction X_{sim}. The classification layer is then used to classify the P and Q into three classes.


Prediction Layer
Aggregation Layer
Matching Layer
Context Representation Layer
Word Representation Layer
Pr(y|P, Q)
softmax
p_1 p_2 ...... p_i ...... p_M
q_1 q_2 ...... q_j ...... q_N

(a) The architecture of BiMPM

(b) The Architecture of Convolutional BiMPM

Figure 3: Architecture of BiMPM and Convolutional BiMPM

3 Experiments

3.1 Environment

Python Version: Python 3.5; GPU: Nvidia GeForce GTX 1080 Ti; CPU: Intel(R) Xeon(R) Gold 6126; Memory: 187GB; Pytorch Version: 0.3.0

3.2 Dataset

The Stanford Natural Language Inference (SNLI) corpus [1]. The SNLI dataset consists of 549367 / 9842 / 9824 (train / valid / test) premise and hypothesis pairs.

3.3 Accuracy Evaluation

The accuracy of these four model is shown in Fig.4. We can find that the BiMPM and CBiMPM model achieve the highese accuracy of over 86%. In the beginning, BiMPM and CBiMPM almost have the same trend. However, after the second epoch, accuracy of CBiMPM become higher. The ConvRNN-Skip also achieve an accuracy over 83%. However, the ConvRNN-Naive method only achieve under 80%.

The hyperparameters are set as: {*"conv": true, "word_dim": 300, "hidden_size": 100, "char_hidden_size": 50, "word_vocab_size": 37252, "data_type": "SNLI", "learning_rate": 0.0007, "kernel_size": 3, "batch_size": 64, "n_fm": 50, "dropout": 0.1, "max_sent_len": -1, "num_perspective": 20, "max_word_len": 27, "class_size": 3, "epoch": 10, "char_vocab_size": 1*}

The best performance of models are shown in Table.1. We can find that the Convolutional BiMPM model achieves the highest performance. This may because the convolutional encoding is effective when capturing the local n-gram information. This method is combined with the Bi-LSTM attention and matching mechanism, which is capable to find the overall feature of sentences. The CBiMPM is a combination of two strong models.

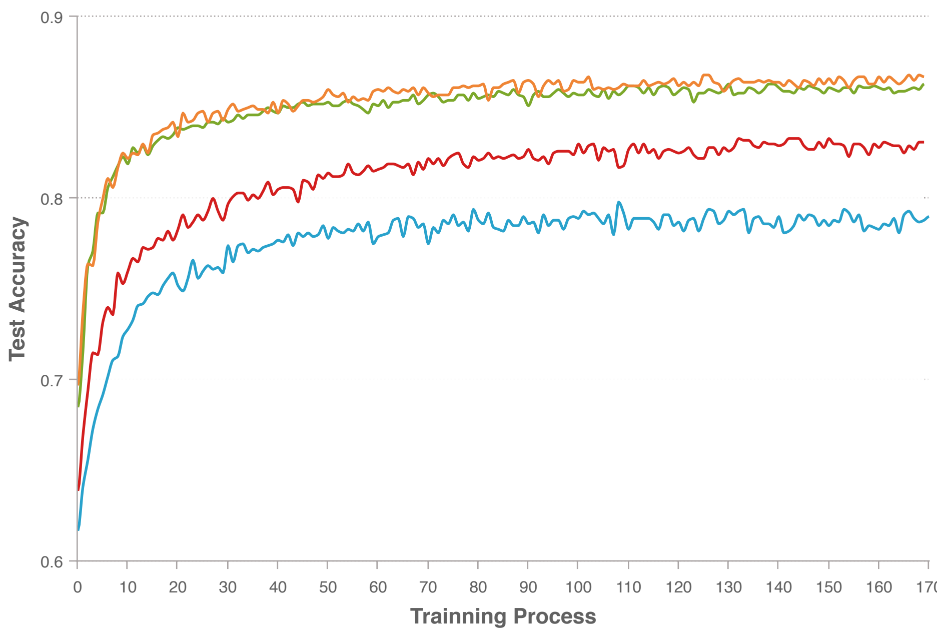


Figure 4: Accuracy in the training process. ConvRNN-Naive: Blue; ConvRNN-Skip: Red; BiMPM: Green; CBiMPM: Orange.

Table 1: Best Accuracy

Model	Best Acc on SNLI
ConvRNN-Naive	79.03%
ConvRNN-Skip	83.31%
DIIN [2]	84.24%
BiMPM [6]	86.49%
CBiMPM	**86.72**%

3.4 Running Time Evaluation

The running time of these four models are shown in Fig.5. All models run for 1 epoch, with other hyper-parameters the same. We can find that the running times of BiMPM and CBiMPM are close. The running times of ConvRNN-Naive and ConvRNN-Skip are close. But the formers cost 3X more seconds than the latters. This is because the parameters of latters are much less.

4 Acknowledgement

The GPU devices used in this project is support by Department of Computer Science, Purdue University.

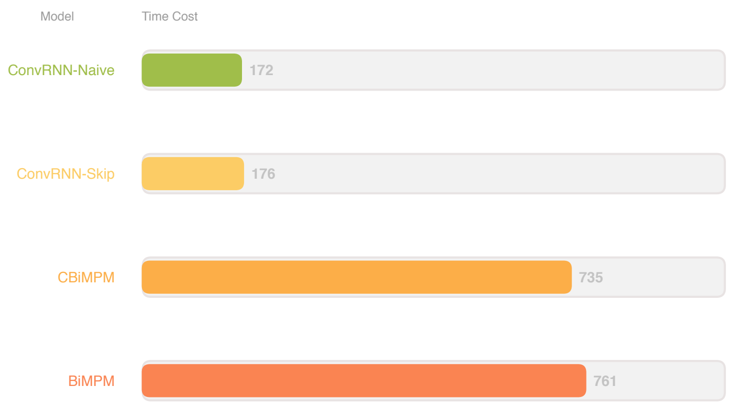


Figure 5: Running Time (seconds) of ConvRNN-Naive, ConvRNN-Skip, BiMPM, and CBiMPM.

References

[1] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[2] Y. Gong, H. Luo, and J. Zhang. Natural language inference over interaction space. *arXiv preprint arXiv:1709.04348*, 2017.

[3] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[4] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[5] C. Wang, F. Jiang, and H. Yang. A hybrid framework for text modeling with convolutional rnn. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2061–2069. ACM, 2017.

[6] Z. Wang, W. Hamza, and R. Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.